Fig. 1

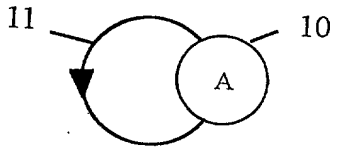


Fig. 2

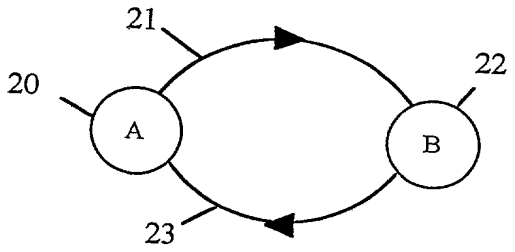
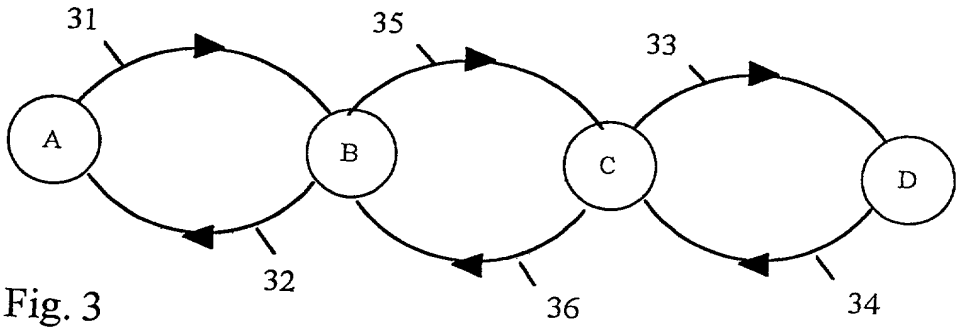


Fig. 3



Translation is not an official translation of the European Patent Office

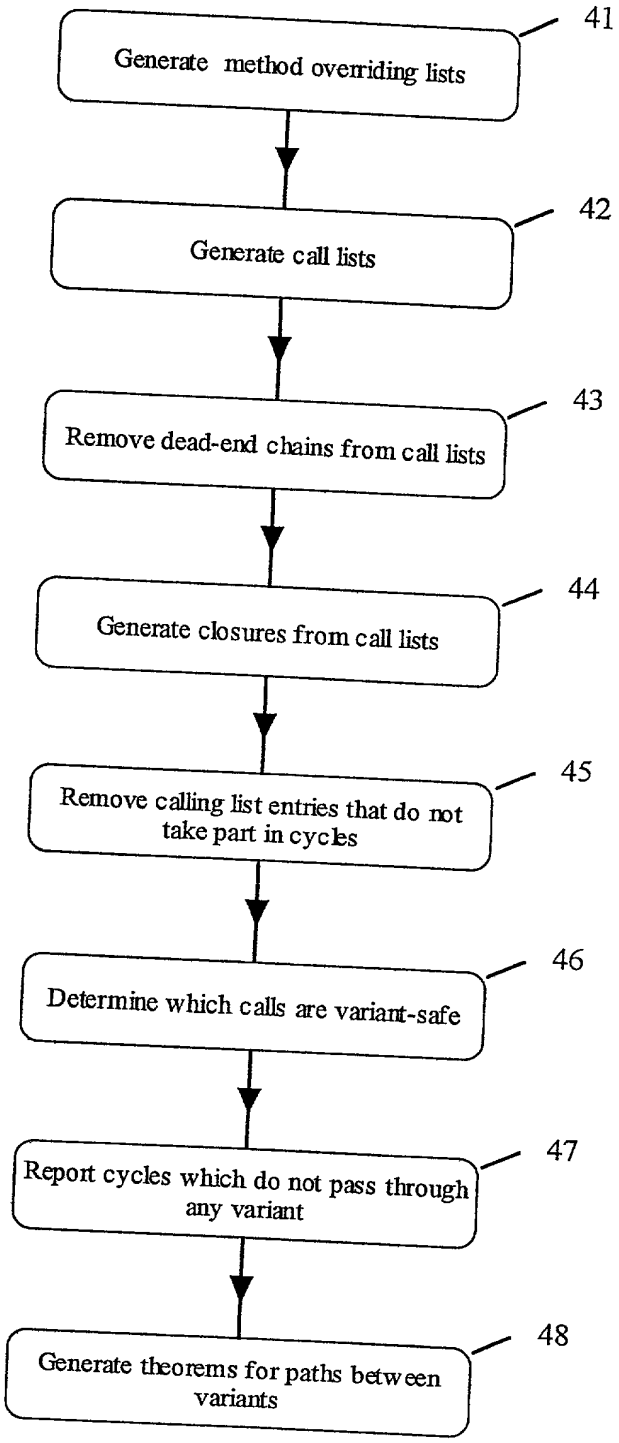

Generate method overriding lists
41
Generate call lists
42
Remove dead-end chains from call lists
43
Generate closures from call lists
44
Remove calling list entries that do not take part in cycles
45
Determine which calls are variant-safe
46
Report cycles which do not pass through any variant
47
Generate theorems for paths between variants
48

Fig. 4

Step 41

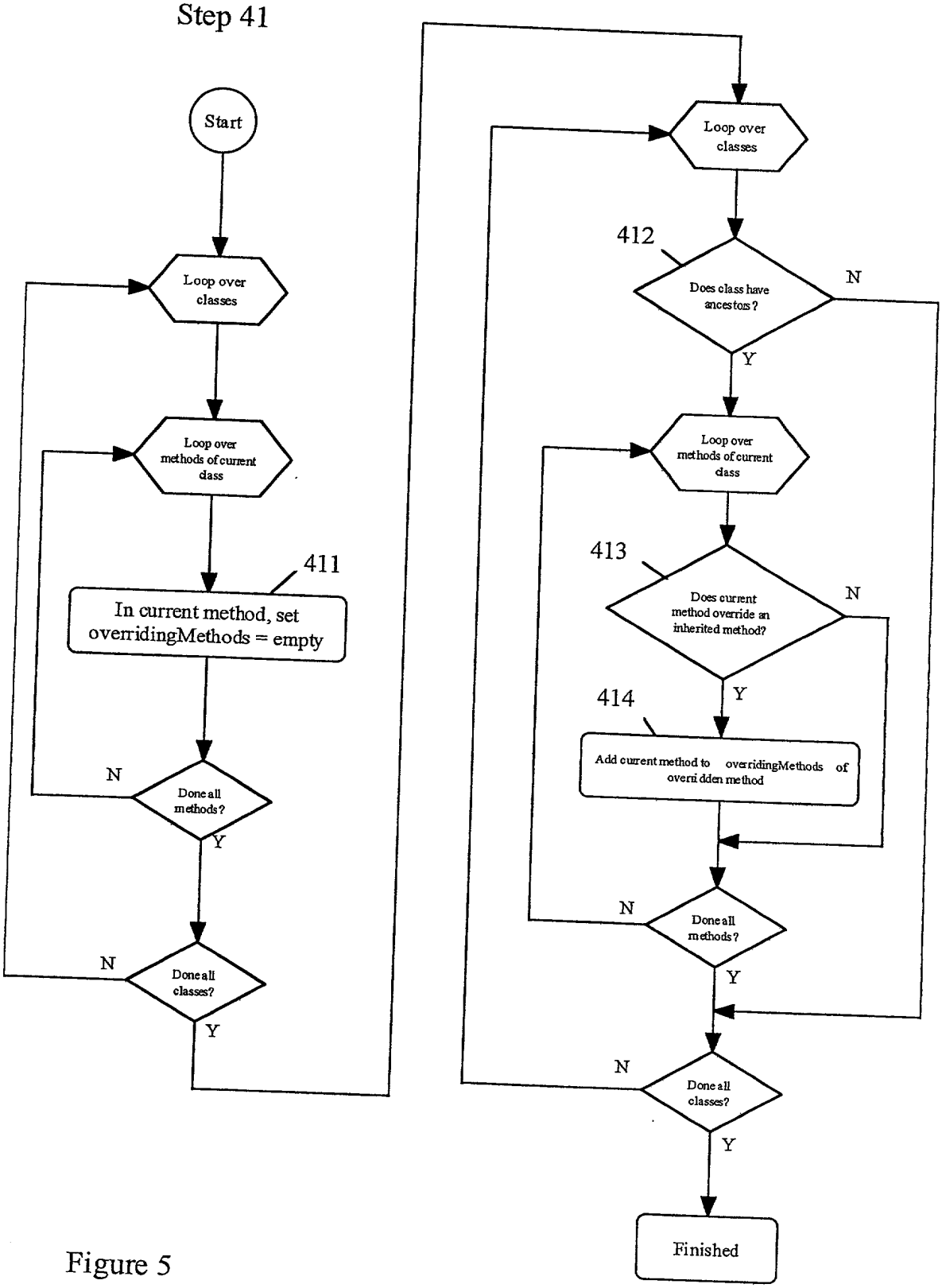


Figure 5

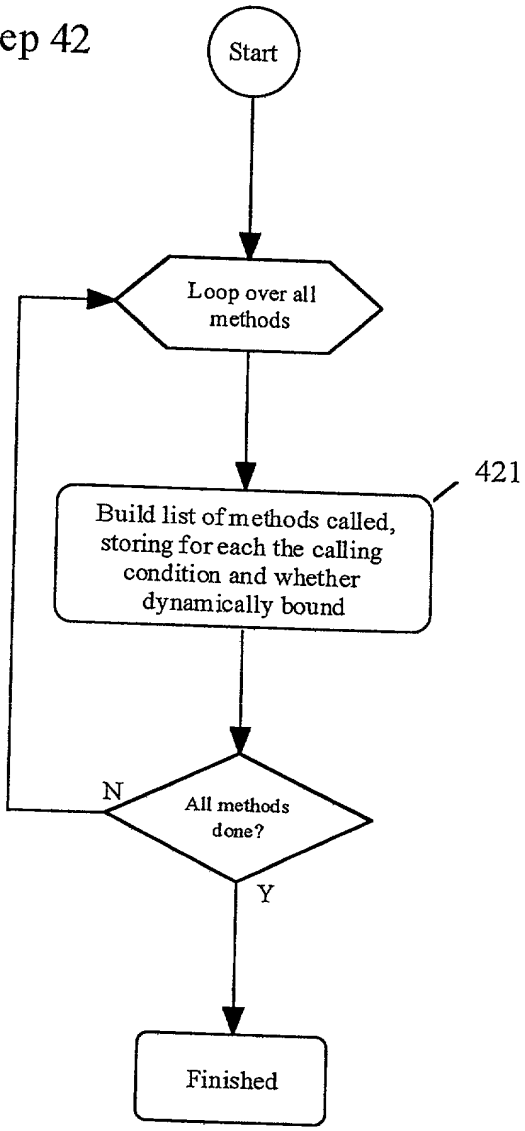

Step 42
Start
Loop over all methods
Build list of methods called, storing for each the calling condition and whether dynamically bound
421
All methods done?
N
Y
Finished

Figure 6

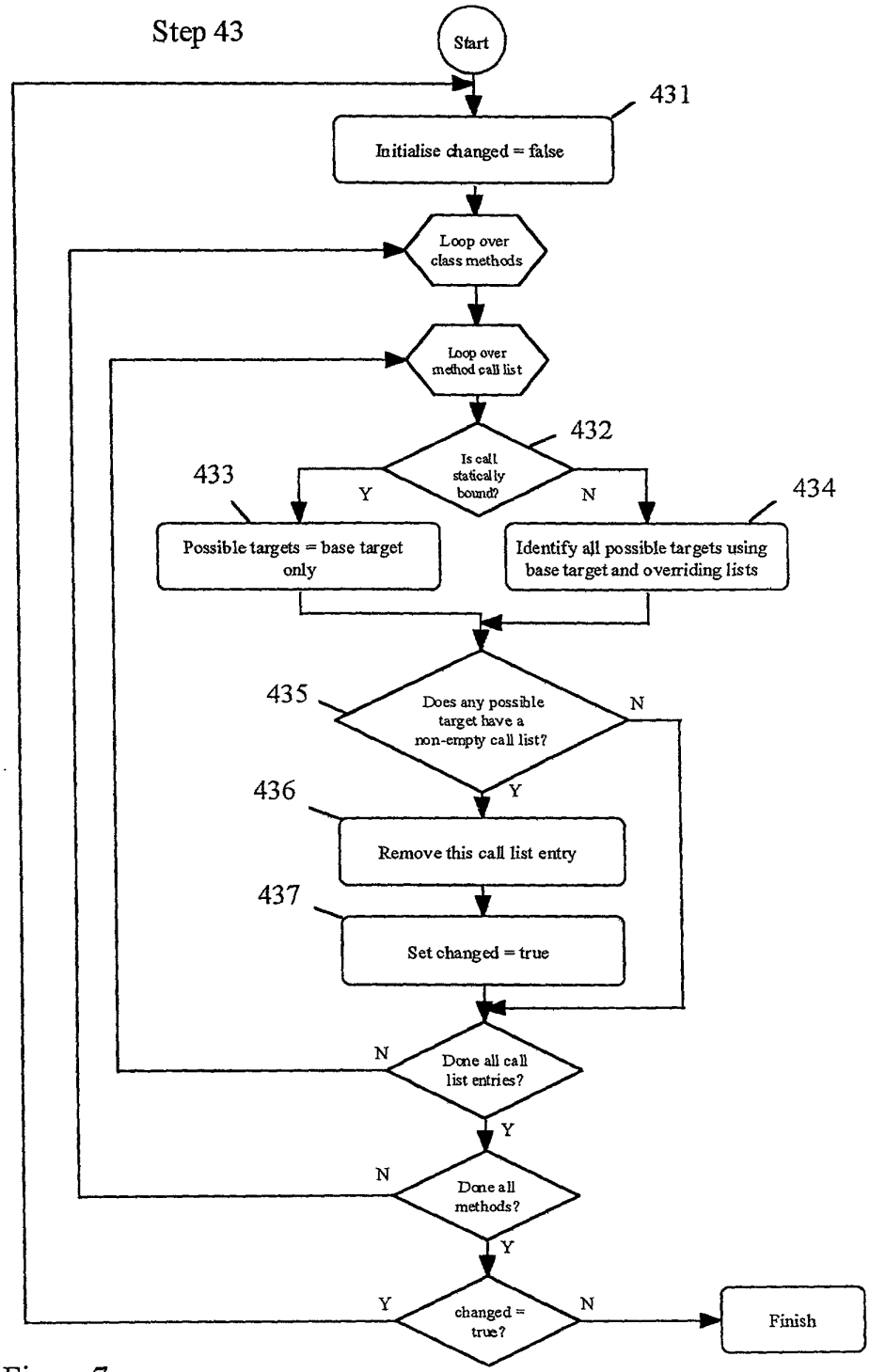

Step 43
Start
431
Initialise changed = false
Loop over class methods
Loop over method call list
432
Is call statically bound?
433
Y
N
434
Possible targets = base target only
Identify all possible targets using base target and overriding lists
435
Does any possible target have a non-empty call list?
N
Y
436
Remove this call list entry
437
Set changed = true
N
Done all call list entries?
Y
N
Done all methods?
Y
Y
changed = true?
N
Finish

Figure 7

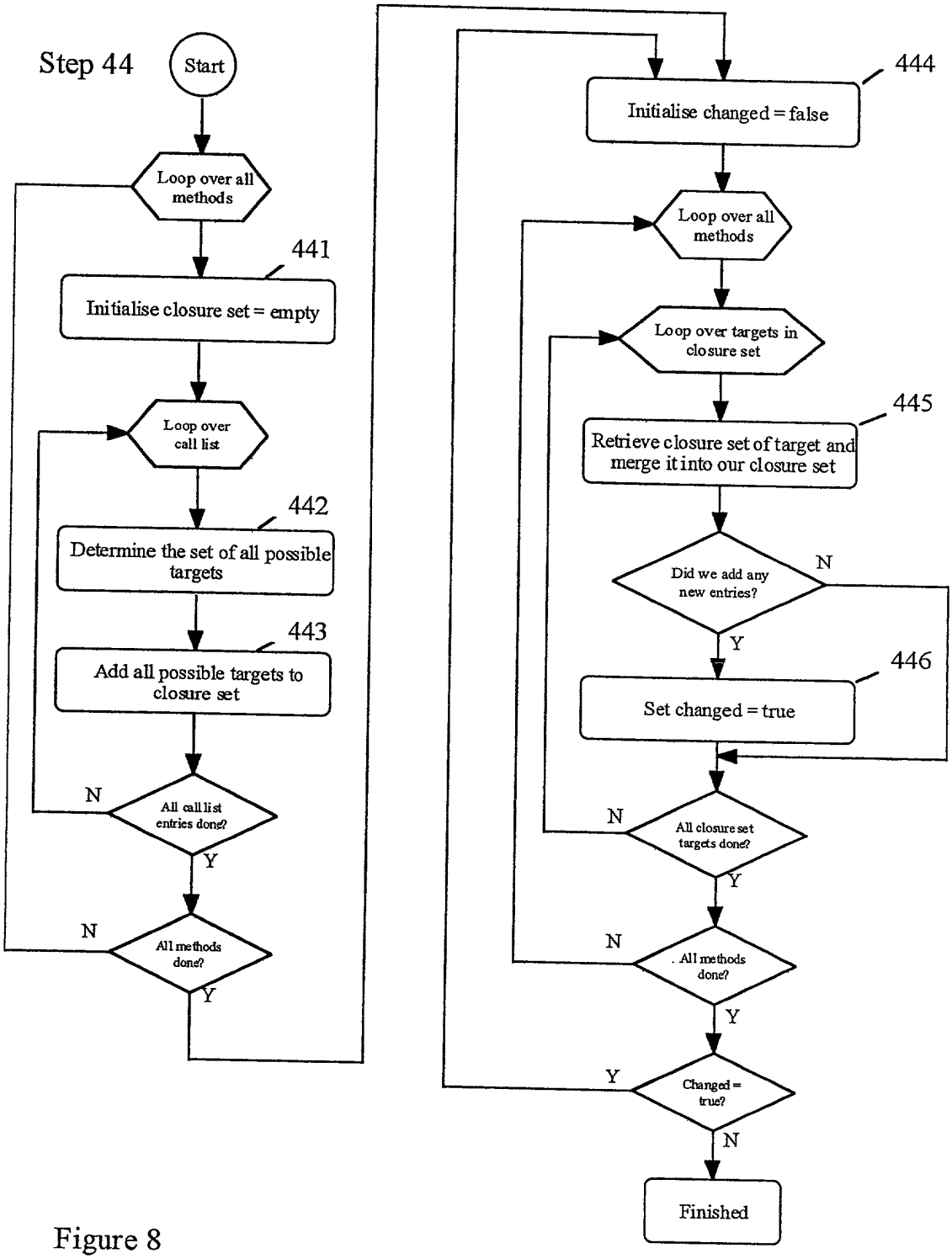

Step 44
Start
Loop over all methods
441
Initialise closure set = empty
Loop over call list
442
Determine the set of all possible targets
443
Add all possible targets to closure set
All call list entries done?
N
Y
All methods done?
N
Y
444
Initialise changed = false
Loop over all methods
Loop over targets in closure set
445
Retrieve closure set of target and merge it into our closure set
Did we add any new entries?
N
Y
446
Set changed = true
All closure set targets done?
N
Y
All methods done?
N
Y
Changed = true?
Y
N
Finished

Figure 8

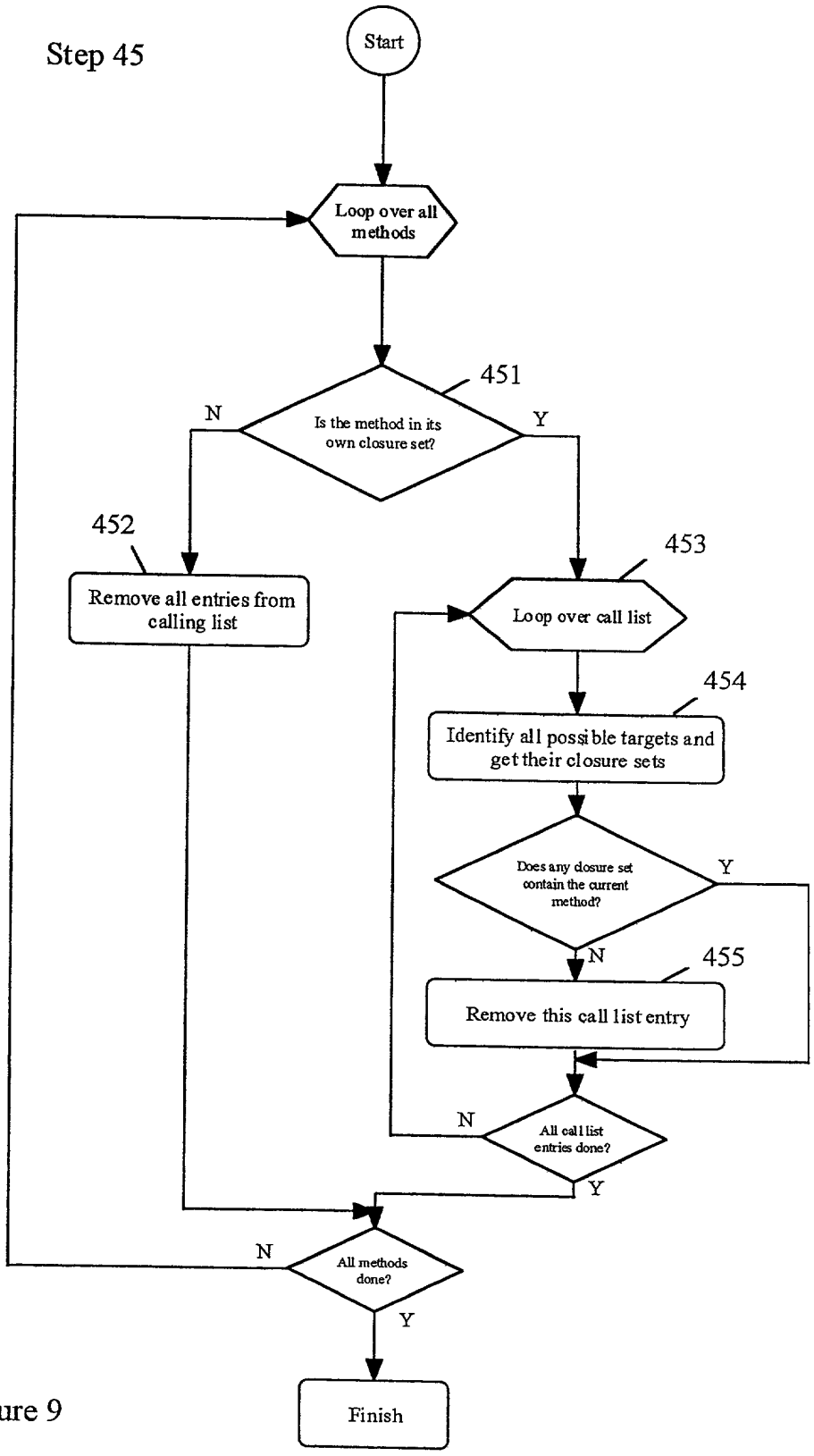

Step 45
Start
Loop over all methods
451
Is the method in its own closure set?
N
Y
452
Remove all entries from calling list
453
Loop over call list
454
Identify all possible targets and get their closure sets
Does any closure set contain the current method?
Y
N
455
Remove this call list entry
N
All call list entries done?
Y
N
All methods done?
Y
Finish

Figure 9

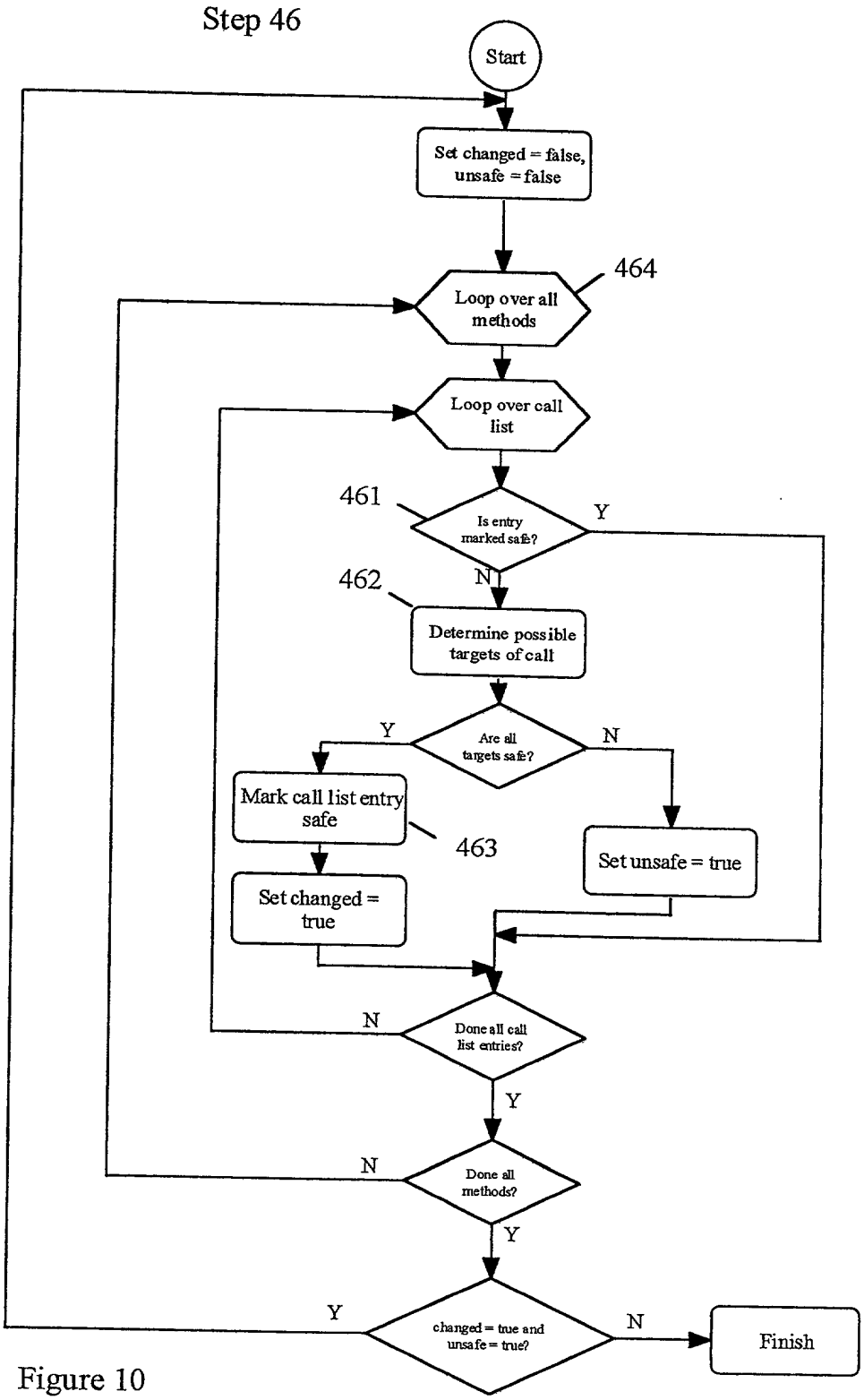

Step 46
Start
Set changed = false, unsafe = false
464
Loop over all methods
Loop over call list
461
Is entry marked safe?
Y
N
462
Determine possible targets of call
Are all targets safe?
Y
N
Mark call list entry safe
463
Set unsafe = true
Set changed = true
Done all call list entries?
N
Y
Done all methods?
N
Y
changed = true and unsafe = true?
Y
N
Finish

Figure 10

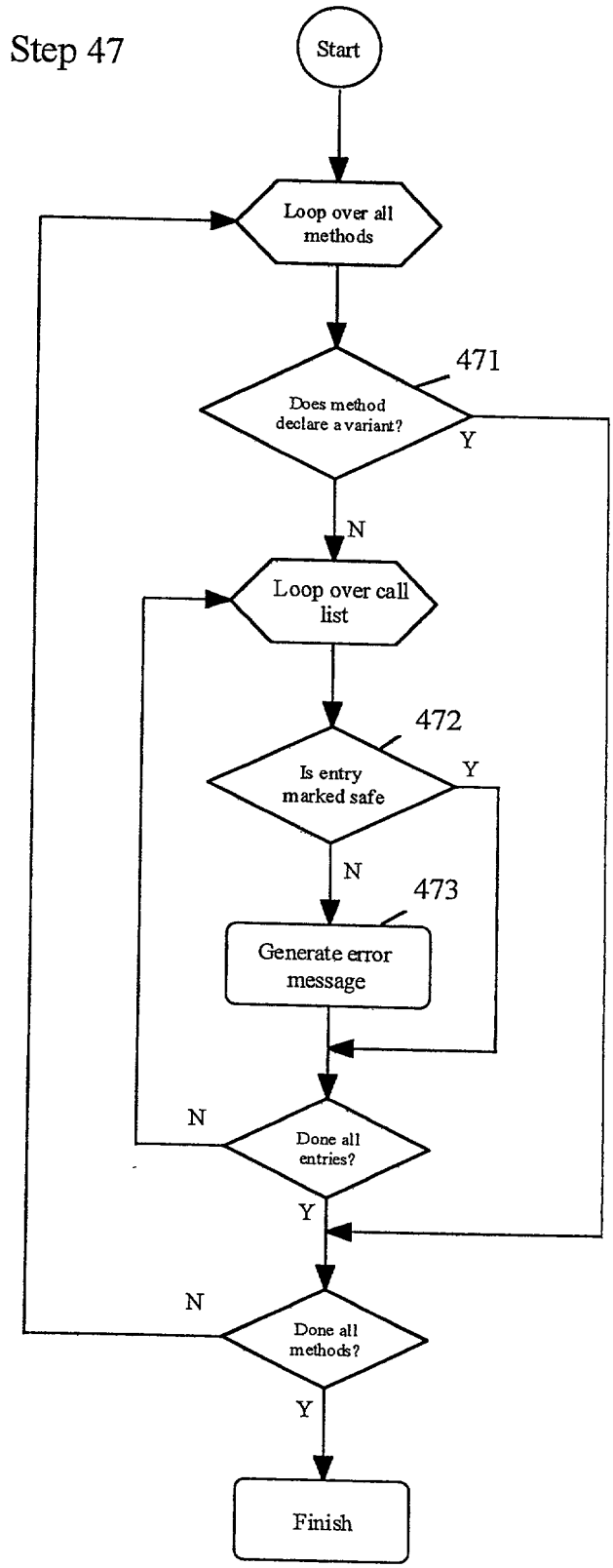

Step 47
Start
Loop over all methods
471
Does method declare a variant?
Y
N
Loop over call list
472
Is entry marked safe
Y
N
473
Generate error message
N
Done all entries?
Y
N
Done all methods?
Y
Finish

Figure 11

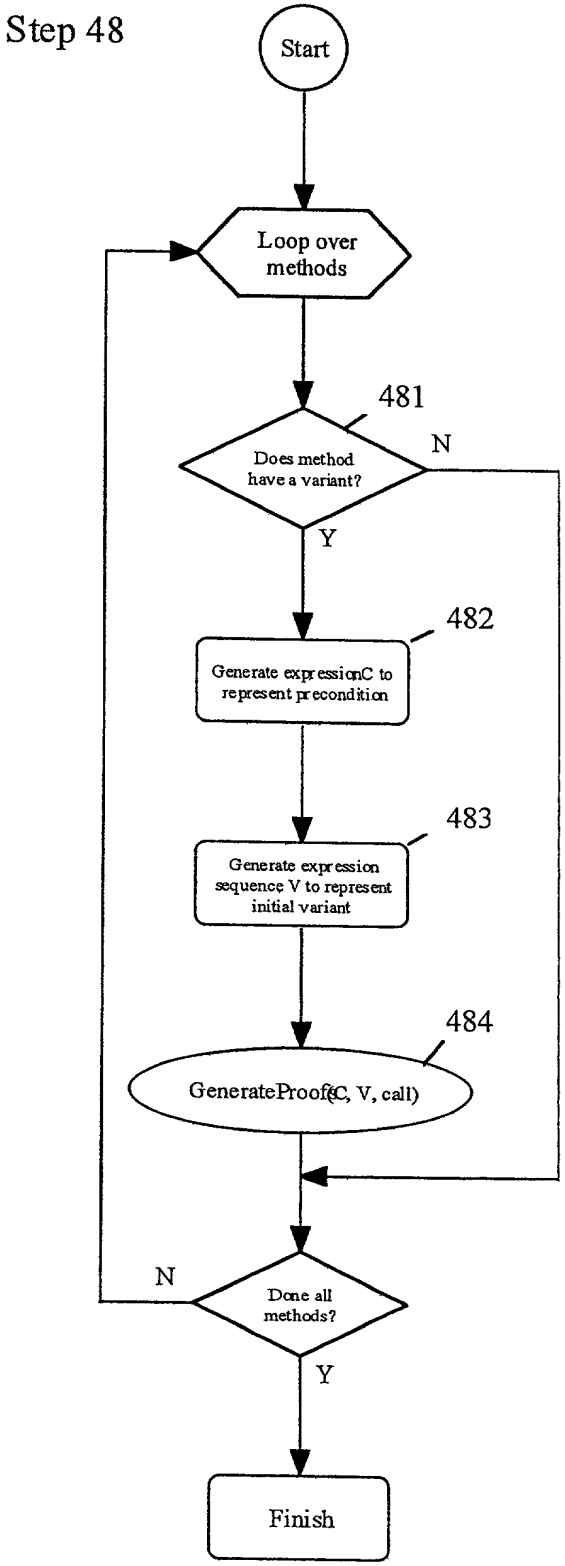

Step 48
Start
Loop over methods
481
Does method have a variant?
N
Y
482
Generate expressionC to represent precondition
483
Generate expression sequence V to represent initial variant
484
GenerateProof(C, V, call)
N
Done all methods?
Y
Finish

Figure 12

Procedure GenerateProofs(C, V, L)

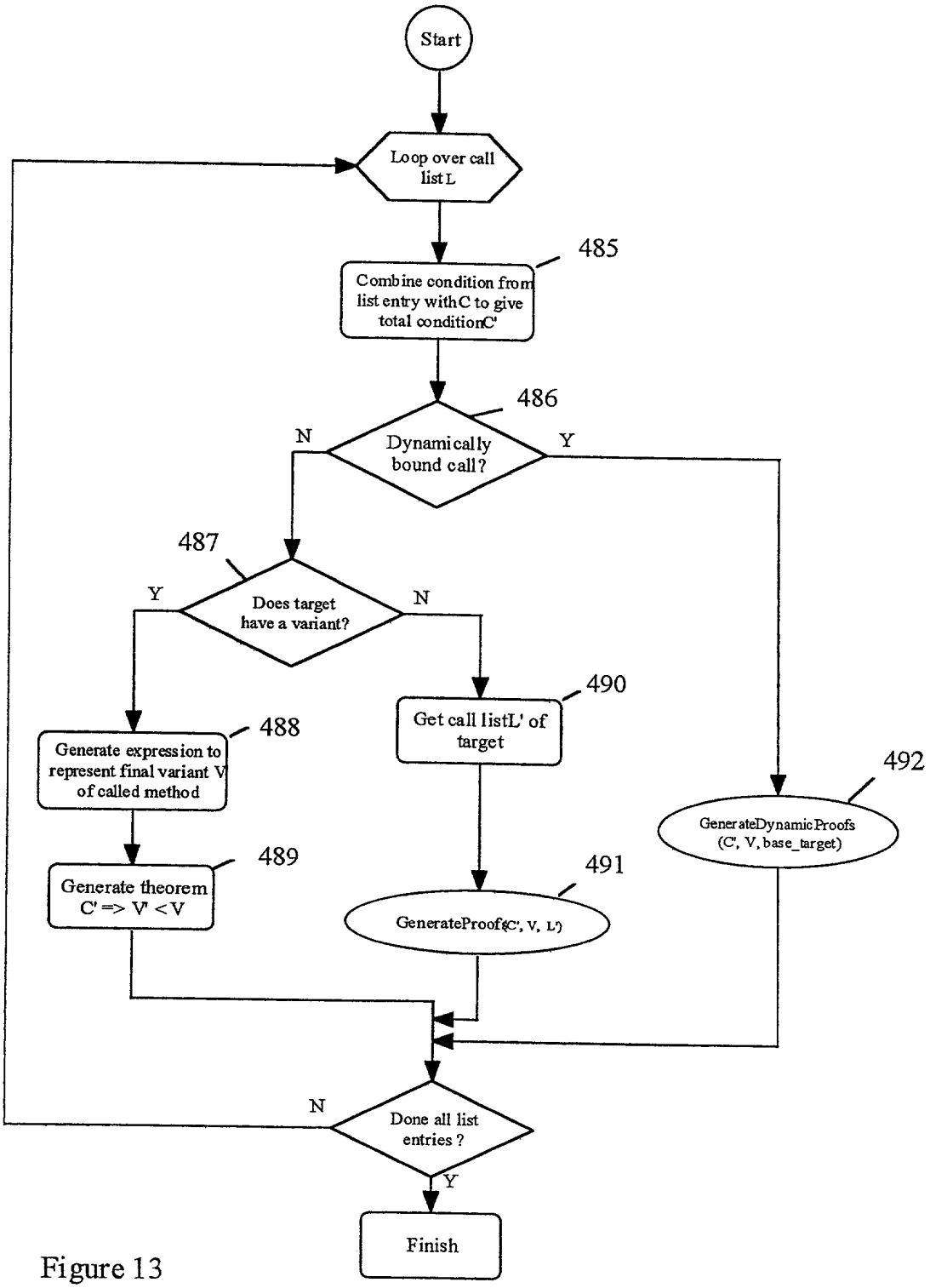


Figure 13

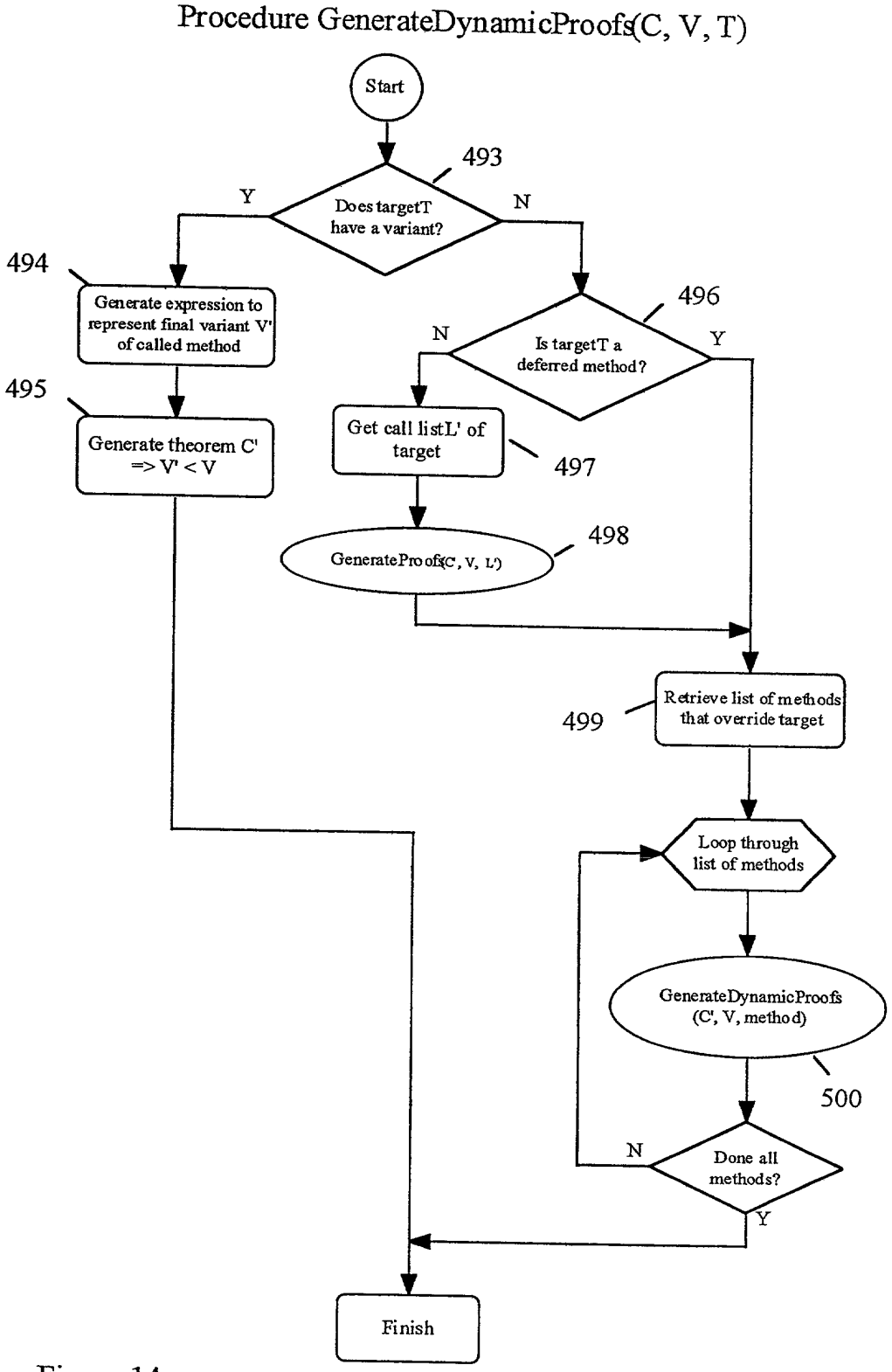

Procedure GenerateDynamicProofs(C, V, T)
Start
493
Does targetT have a variant?
Y
N
494
Generate expression to represent final variant V' of called method
495
Generate theorem C' => V' < V
496
Is targetT a deferred method?
N
Y
Get call listL' of target
497
GenerateProofs(C, V, L')
498
499
Retrieve list of methods that override target
Loop through list of methods
GenerateDynamicProofs (C', V, method)
500
N
Done all methods?
Y
Finish

Figure 14